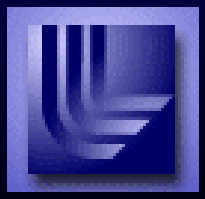
PICA: Parallel Image Compositing API Overview

Randall Frank

Lawrence Livermore National Labs

UCRL-PRES-201691

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.



Scalable Visualization: Compositors

"Standard" commodity clusters

- Nodes,GPU,Interconnect + PowerWalls + Remote displays

Multiple goals, but fundamentally aggregation

- Interaction/VR: High frame rates
- PowerWalls: Large pixel counts
- Data scaling: High polygon/fill rates
- Image Quality: Full scene anti-aliasing

Hardware for "compositing" focused on:

- Application transparency
- Parallel rendering models

Compositing solutions vary considerably:

- HP sv6/sv7, Lightning2, Sepia, SGE, Orad, SGI multi-pipe SDK
- Both transparent and opaque APIs

Inhibitors to Compositor Adoption

Small market, How many clusters out there?

- No such thing as a “standard” cluster
- Most clusters do not have physical compositor solutions

Few applications

- No common rendering infrastructure for parallel apps
- No common API for compositors
  - Application transparent modes are scalability limited
  - Invasive/custom interfaces to devices for “special features”
  - “Expensive” for ISVs to support
- Lack of 2D integration with 3D
  - Application interaction models change
- ASCI scalable rendering stack targets this

Hardware limitations exist

- Capabilities of COTS graphics cards

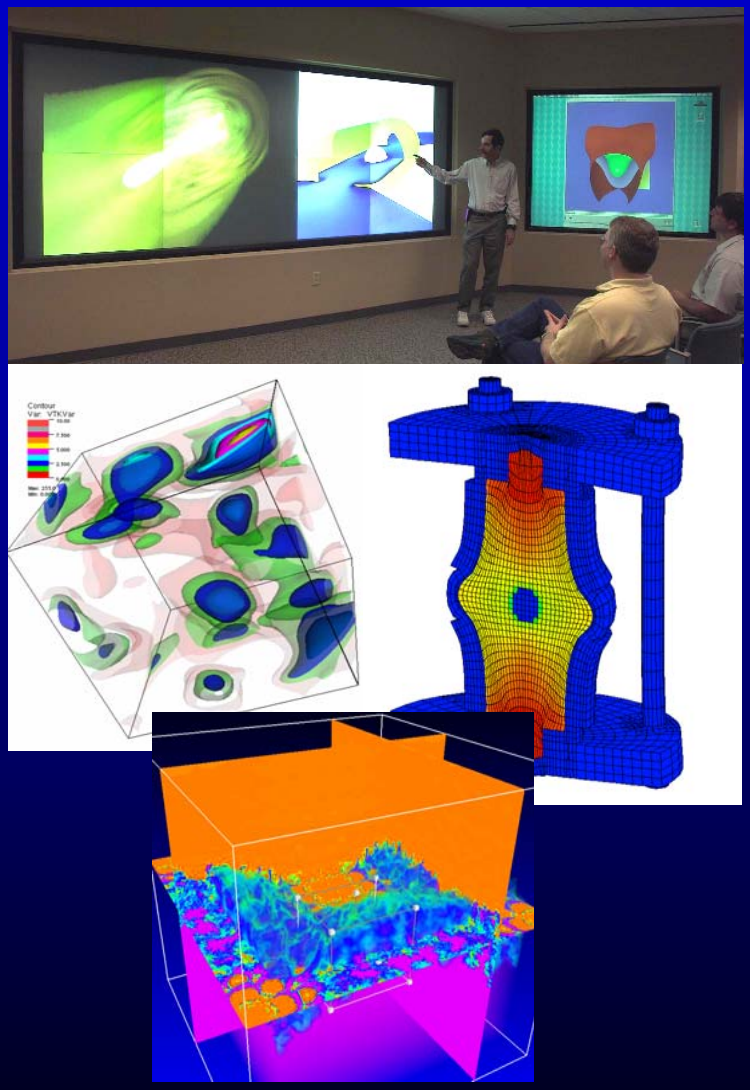
Parallel Image Compositing API (PICA)

System developers and early adopting app developers

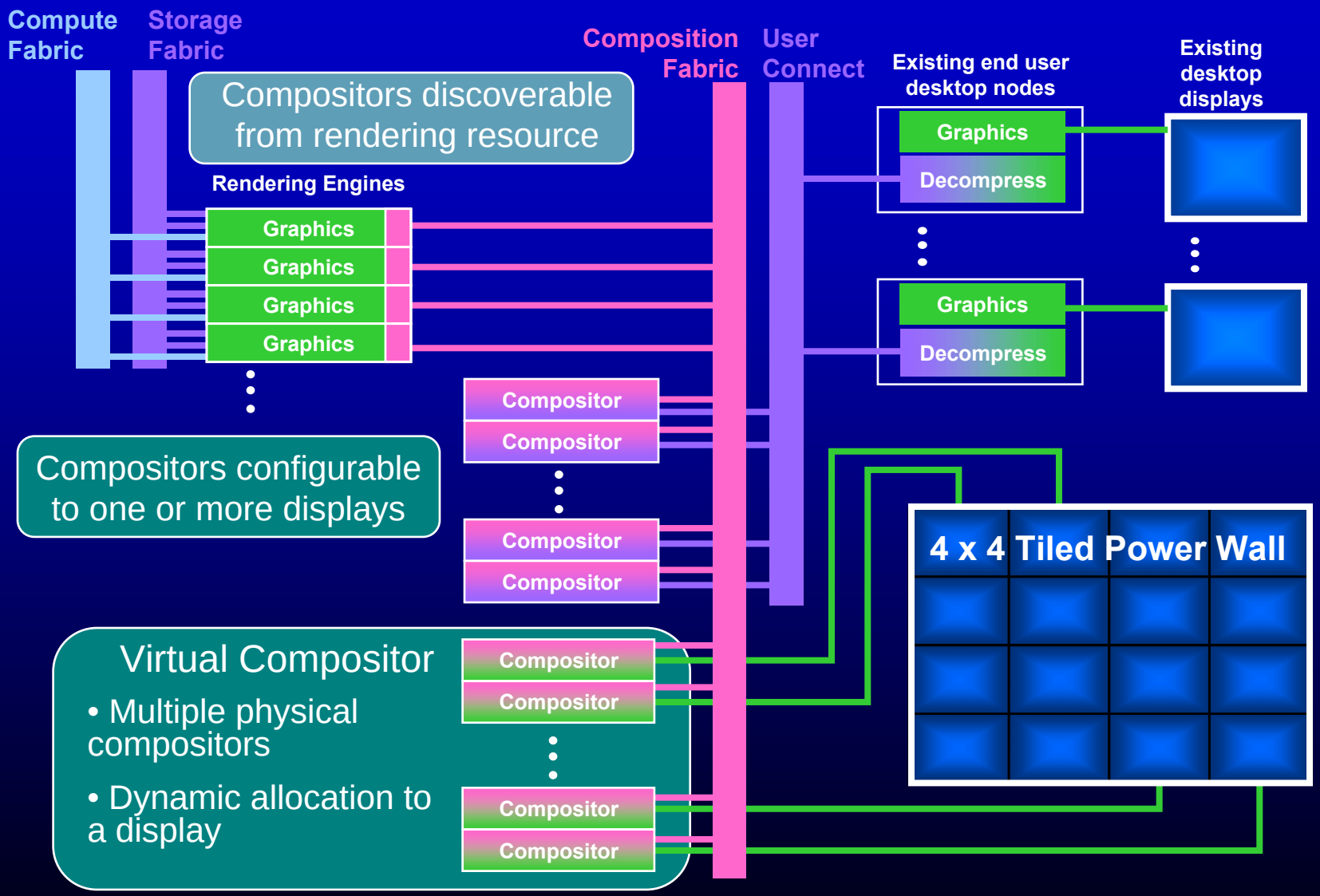
- Mostly HP, PNNL and LLNL employees
- Informal API discussions

Goals

- Abstraction for distributed image composition, both 2D and 3D
- Provide an open source API adoptable by ISVs
- Provides a platform for SW implementations
- Support major hardware compositors
- Allow for optimal implementations
  - Maximize parallel opportunities
  - Avoid explicit collective operations



An Idealized Visualization Environment



Underling Assumptions

API targets parallel applications

- Simple parallel model assumptions
- Application can pass messages to itself

API must handle various input sources

- Region of graphics card memory
- Software rendering to main memory
- Must support “windowed” applications

API must provide a complete compositing abstraction

- Compositing function pipeline covering common usage
- Abstract the concept of composite “ordering”

Independent of graphics API

- DirectX, OpenGL, Custom renderer

Must abstract all current compositor forms

- Multiple compositors available in the same cluster
- DVI based, network based, software, etc

Basic PICA Abstractions

Application "Nodes"

- A source of image framelets, includes rendering resource
  - Not the same as a cluster node
- Supports a partition of the application execution

Application generates a sequence of "Frames"

- Frames are sequenced by IDs
- Limited queries Supported via frame IDs
- Multiple frame "channels" for stereo

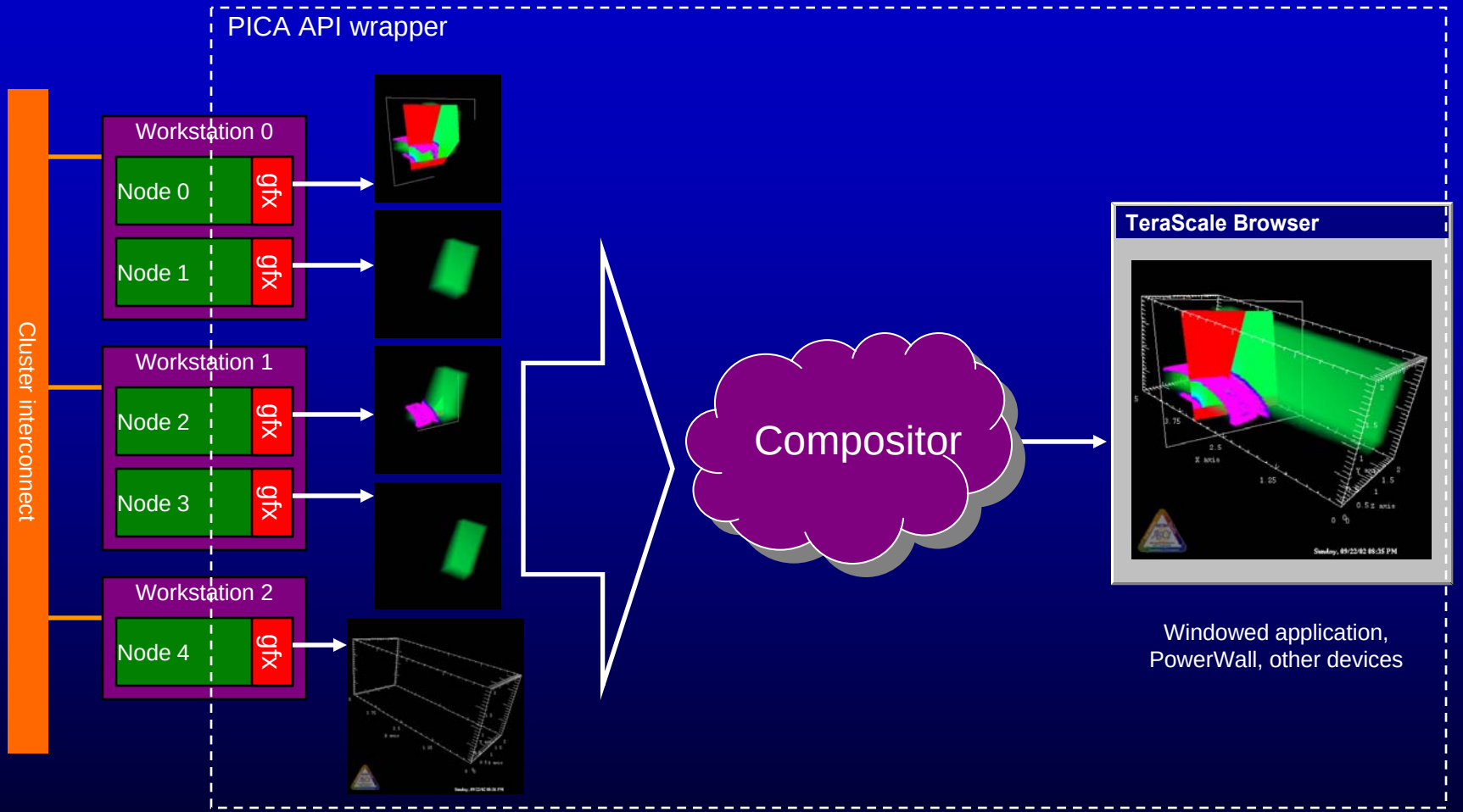
Frames are built from multiple "Framelets"

- An framelet is a (rendered) rectangle of augmented (e.g. α & depth) pixels
- Individual nodes can submit multiple framelets
- Framelets can be located anywhere in a frame
- Framelets are tagged with an "order" number within each frame

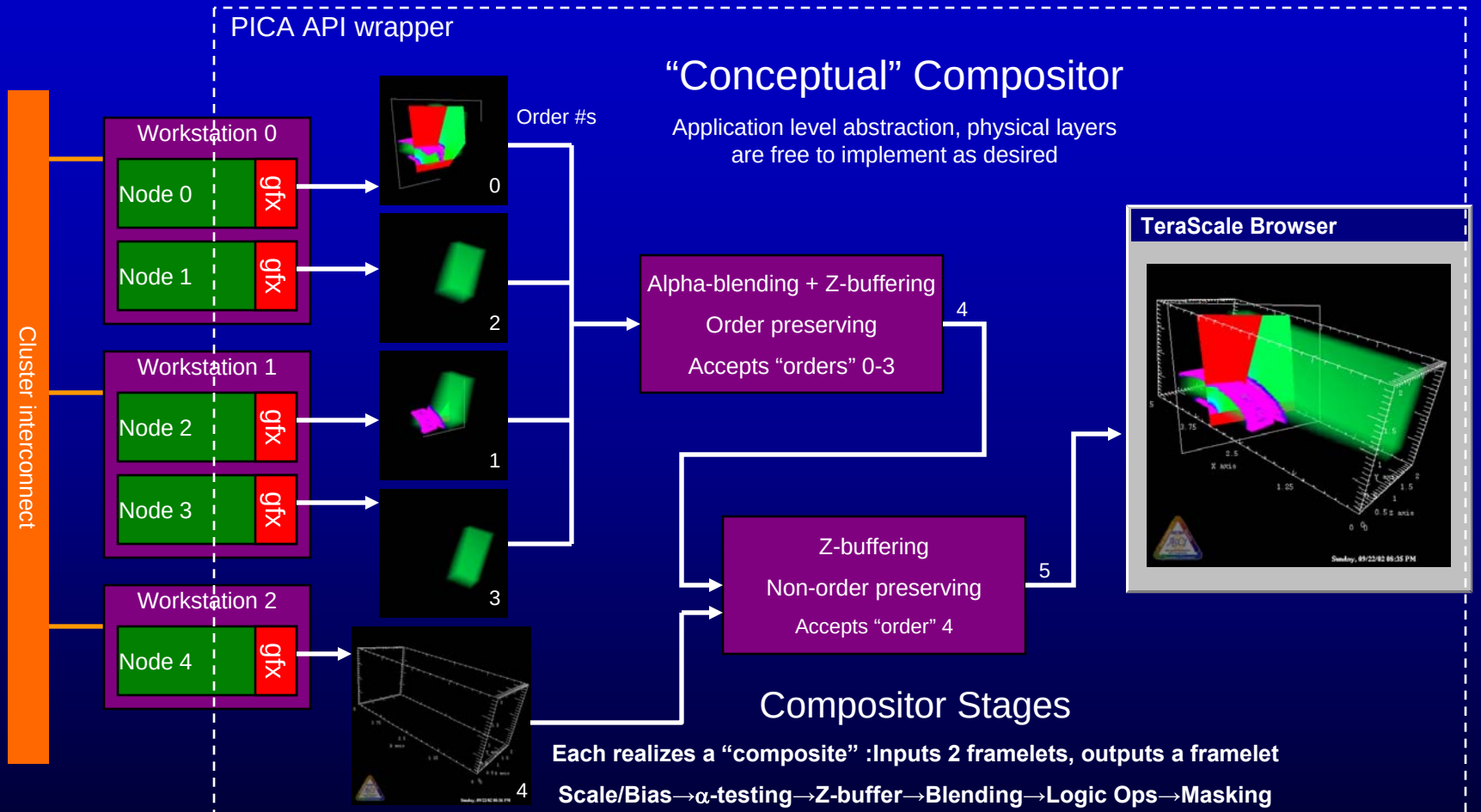
"OpenGL"-style compositing pipeline

- Multiple conceptual "stages" of OpenGL compositing supported
- Framelet "order" values specify the route through the stages

Compositing Operation in PICA



Compositing Operation in PICA:Details



Nodes	Framelets	Context	Frame
Generate imagery	Ordered image rects	Realization of a compositor	Output complete images

Basic PICA Operation

initialization

Compositing "context" negotiated (one node)

- Includes compositing pipeline definition
- Hardware is allocated

Context ID is passed to nodes by application

Context is "bound" locally (all nodes) to realize the system

per-frame operation

Application starts a frame (all nodes)

- Application renders graphics (generates framelets)
- Framelets are passed to local context

Application ends the frame (all nodes)

- Composite may occur asynchronously

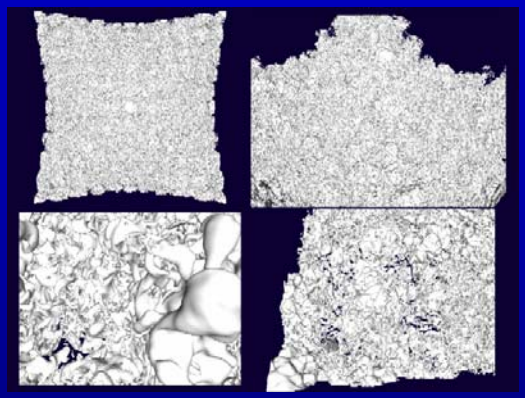
Basic query functions allow for application feedback

Technical Details

Compositing at application rates, not display rates

Designed to support 99% of applications

- Sorted, alpha blending (e.g. volume rendering)
- Tiling
- “Overlays” (e.g. annotations, heads-up-displays)

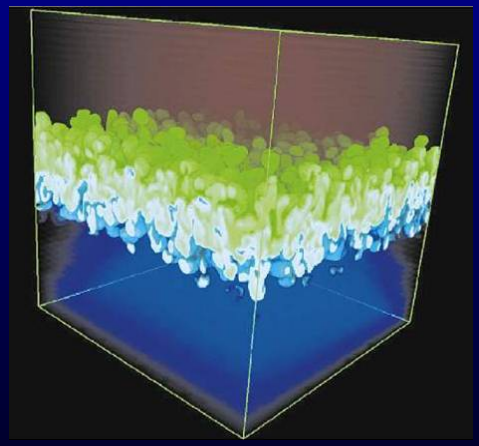


Advanced composites supported as well

- Full-screen anti-aliasing
- HW assisted “transparency”

Mechanisms for application “hinting”

- Performance optimizations (e.g. composite trees)
- Information regarding the degree of HW assist
- Negotiation of “special features”



Additional Supported API Features

"Negotiated" compositor features

- Incomplete compositor crossbar
  - Local order constraints
- Non-volatile frame buffers
  - Incremental/frameless rendering w/copyswap devices
- Window system "aware", buffer & "wall" mode rendering
- Dynamic composite functions, framelet scaling

Integration with Chromium

- Cr lacks sufficient information to exploit HW
  - Need standard way of exposing this to applications
- Portions of PICA being adopted as the Cr interface
- Cr as optimization layer, PICA as realization for unaware apps

Current Status

Second revision of the specification is complete

- In conjunction with Brian Paul, Tungsten Graphics
- “Human readable” specification next
- Basic startup on both SMP and distributed systems done

Development efforts

- “C” Stubs written for the API
- Compiles both apps and compositor
- Tiered shared library dispatch done
- Simple software compositor under development
- Test application under development

Future Steps: Making it Real...

USDC proposal by Tungsten Graphics

- Provide “RedBook” level documentation
  - Get API feedback from a wider audience
    - Researchers
    - Application writers
    - HW providers
  - Document expected usage models
- Create 1.0 sample implementation
  - Develop complete, reusable, SW version
  - Write drivers to support a HW compositor

Consider collected issues for version 2.0

Auspices:

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.